Python PANDAS Visualisation Cheat sheet
V1 date: 11/11/2024
Library Seaborn version 0.13.2

Plotting general

Import seaborn and matplot (to show the plots) libraries:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

The following table shows data types. For some you can create summary statistics. The best visualisation depends on the type of data and research question. Ensure accurate labels, a meaningful title, and distinct colours to ease understanding.

Qualitative / categorical data	nominal	Cannot be measured. The data can be grouped. Then frequency or percentage distribution can be calculated and visualised. For example: hair colour, nationality, gender, symbols
	ordinal	Follows a natural order, however the distance between data values is not exact. For example: customer satisfaction, educational levels, level of happiness.
Quantitative / numerical data	discrete	Finite number of values with an equal distance between each value. Summary statistics such as median, mean, etc. can be calculated. For example: number of participants in a meeting, number of chocolate bars
	continuous	Has an infinite number of values within a range. Summary statistics can be calculated. For example: temperature range

The seaborn library includes sample data sets that can be utilised for practice or showcase of knowledge. The following charts are based on the tips data set:

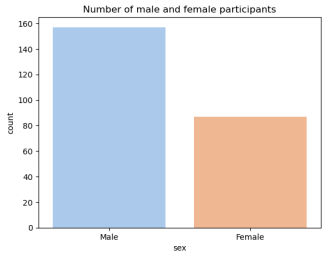
```
tips = sns.load_dataset("tips")
```

Bar chart

Use case: nominal data, ordinal data, discrete data, continuous data
Easy to understand, see 2nd page for an example of a stacked bar chart.

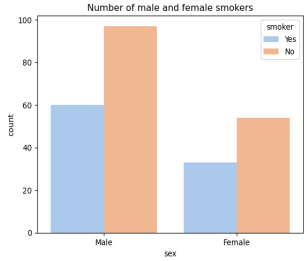
```
sns.countplot(tips, x="sex")
```

Count bar chart sex



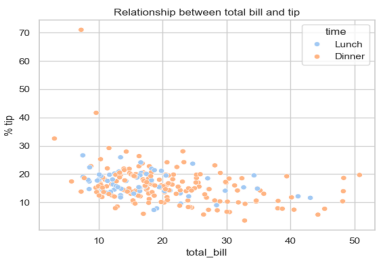

```
sns.countplot(tips, x="sex",
hue="smoker")
```

Count bar chart sex / smoker y/n



Scatter plot

Use case: continuous data
X-asis independent variable, y-axis dependent variable. An independent variable is not changed by the other variables or independent variables affect dependent variables. Scatter plots show the relationship between variables. Good to show correlation, trends, and outliers.




```
tips['% tip'] =
(tips['tip']*100)/tips['total_bill']
sns.scatterplot(data = tips,
                x = 'total_bill',
                y = '% tip',
                hue = 'time',
                palette = 'pastel');
plt.title('Relationship between total bill and tip')
plt.show()
```

Facet grid map

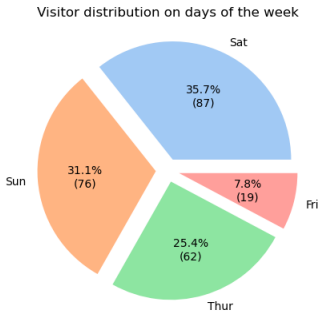
Use case: nominal data, ordinal data, discrete data, continuous data
Forms a matrix of plots / charts, for example, perfect to explore the relation ship between two discrete variables, or one discrete variable and a continuous variable. Can show a matrix of many chart types, herein a scatterplot with regression line.

```
tips['% tip'] = (tips['tip']*100)/tips['total_bill']
p = sns.FacetGrid(tips, col = "time", hue = "smoker", palette="pastel")
p.map(sns.regplot, "total_bill", "% tip").add_legend()
p.set(xlabel='Total bill', ylabel="Tip as %age of total bill")
plt.show()
```

Pie chart

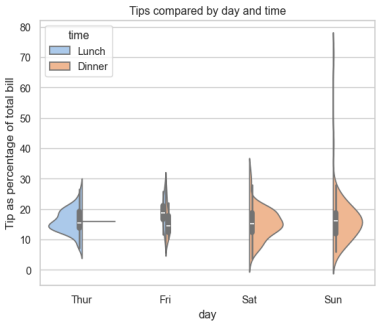
Use case: nominal data, ordinal data, discrete data, continuous data
Good to visualize and compare proportions and percentages in a group of data. Does not show change over time.

```
def p_form(values):
        def label(pct):
            total = sum(values)
            p =
int(round(pct*total/100.0))
            return '{:.1f}%\
n({v:d})'.format(pct, v=p)
        return label
dcount = tips['day'].value_counts()
colors = sns.color_palette('pastel')[0:5]
explode = [0.1, 0.1, 0.1, 0.1]
plt.pie(dcount, labels = dcount.index,
autopct=p_form(dcount), colors = colors,
explode=explode)
plt.title('Visitor distribution over days
of the week')
plt.show()
```



Violin plot

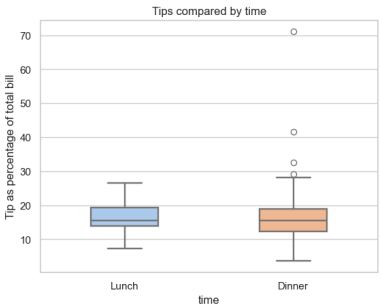
Use case: discrete data, continuous data
Similar to a box plot for statistical visualisation. The white dot in the middle is the median, the ends of the black box the show the lower 25%tile (low) and upper 25%tile (up), the whiskers show the first quartile -1.5 IQR (low) and third quartile + 1.5 IQR (up). Density estimation presents the outline, its extension below or above the whiskers represents outliers in the data. It also shows data distribution, herein, data for lunches and dinners is not available for each day.



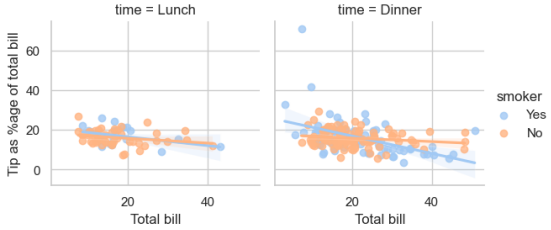

```
tips['% tip'] = (tips['tip']*100)/tips['total_bill']
sns.violinplot(x="day", y="% tip", hue="sex",
                    data=tips, palette="pastel", split=True,
                    scale="count")
plt.title('Relationship between day, %age tip, sex')
plt.show()
```

Box plot

Use case: ordinal data, discrete data, continuous data
Visualize statistics, good to see outliers, variance, and median. The line in the box presents the median, the boxes upper edge presents the upper 25%tile and lower edge the lower 25%tile. The lower whisker present the first quartile - 1.5 IQR and the upper one the third quartile + 1.5 IQR. Outliers are circles below or above the whiskers.




```
tips['% tip'] = (tips['tip']*100)/tips['total_bill']
sns.set_theme(style="whitegrid")
sns.boxplot(data = tips, x = "time", y = "% tip", width = 0.4,
palette = "pastel", linewidth = 1.7).set(ylabel='Tip as percentage of
total bill')
plt.title('Tips compared by time')
plt.show()
```



Pivot table - plot

Pivot tables are a neat method to obtain summary insights from the data. Pivot_table() reshapes the data, transforming rows into columns, and calculates aggregate summary statistics such as sums, counts, averages, etc. (see cheat sheet PANDAS DataFrame). By adding .plot behind the table, e.g. show a bar chart of the data import dataframe_image as dfi

```
tips = sns.load_dataset("tips")
tips['% tip'] = (tips['tip']*100)/tips['total_bill']
tips_piv = np.round(pd.pivot_table(tips, values=['total_bill', 'tip'],
                                   index=['day'],
                                   aggfunc=np.mean,
                                   fill_value=0),2)
tips_piv=tips_piv.reindex(tips_piv['total_bill'].sort_values(ascending=False).index).nl
argest(10, 'total_bill')
df_styled = tips_piv.style.format({'total_bill':'${0:,.2f}',
                   'tip':'${0:,.2f}'}).bar(color='#d65f5f')
df_styled.set_caption("Mean tips and mean total_bill for all sampled days")
dfi.export(df_styled, '/home/user/Data/Blog/# Data Analysis/df_styled.png')
```

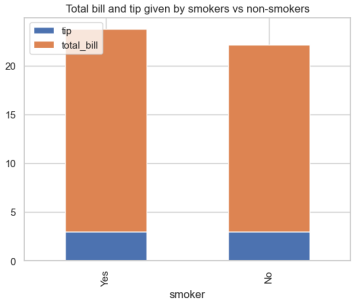
Mean tips and total_bill for all sampled days

day	tip	total_bill
Sun	$3.26	$21.41
Sat	$2.99	$20.44
Thur	$2.77	$17.68
Fri	$2.73	$17.15

Stacked bar chart

Stacked bar chart is a bar chart with each bar showing the related part of a categorical variable. E.g. show the total bill and % tips for smokers and non-smokers in one plot

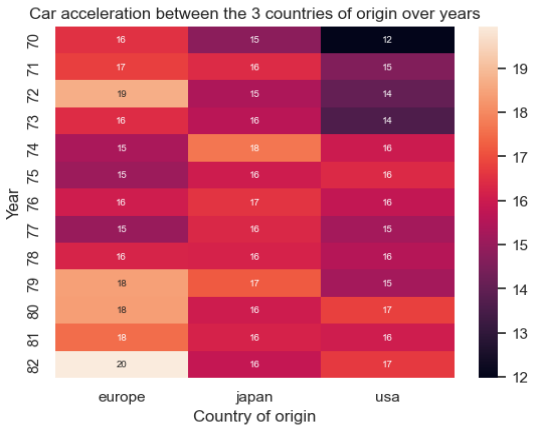
```
tips = sns.load_dataset("tips")
# generate pivot table with summarised values
temp = pd.pivot_table(tips, values=['total_bill', 'tip'], index='smoker')
# print summary table as stacked bar chart
temp.plot(kind='bar', stacked=True)
plt.legend(loc='upper left')
plt.show()
```



Heatmap

Heatmaps are effective for displaying data density or intensity compared in categories or time periods, here years and countries of origin. It is useful for identifying patterns or correlations.

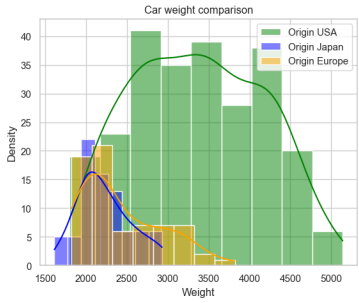
```
mpg = sns.load_dataset("mpg").pivot_table(index="model_year", columns="origin",
values="acceleration", aggfunc='mean')
p1 = sns.heatmap(mpg, annot=True, annot_kws={"size": 7}, cmap="rocket")
p1.set(xlabel="Country of origin", ylabel="Year")
plt.title('Car acceleration between the 3 countries of origin over years')
plt.show()
```



Joint distributions

For comparison of two or more distributions.

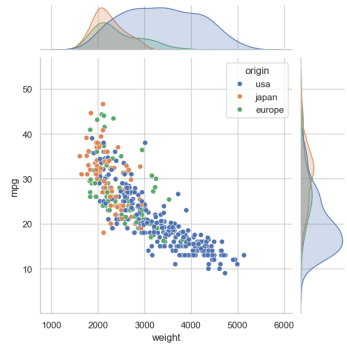
```
data1 =
sns.load_dataset('mpg').query("`origin`
== 'japan'")['weight']
data3 =
sns.load_dataset("mpg").query("`origin`
== 'usa'")['weight']
sns.histplot(data=data3, color='green',
alpha=0.5, kde=True, label='Origin
USA')
sns.histplot(data=data1, color='blue',
alpha=0.5, kde=True, label='Origin
Japan')
plt.xlabel('Weight')
plt.ylabel('Density')
plt.legend()
plt.title('Car weight comparison')
plt.show()
```



Joint plot

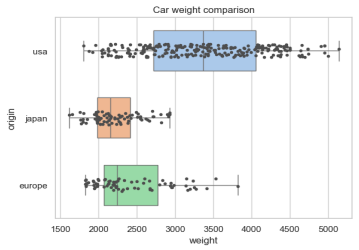
Joint plot comparing mpg and weight of cars between countries of origin.

```
sns.jointplot(data =
sns.load_dataset("mpg"), x =
"weight", y = "mpg", hue =
"origin")
```



Horizontal boxplot with observations

```
# to switch between horizontal and
vertical just switch variables for x
and y in the plot
sns.boxplot(
    data = sns.load_dataset("mpg"),
x="weight", y="origin", hue="origin",
    whis=[0, 100], palette="pastel")
sns.stripplot(sns.load_dataset("mpg"),
x="weight", y="origin", size=4,
color=".3")
plt.title('Car weight comparison')
```



Pair plot

A pairplot compares variables in a dataset pairwise. This overview of the data makes it easier to visualise and understand large data sets.

```
sns.pairplot(tips, hue="sex")
```